CHAPTER 13

Access Levels

Every class member has an accessibility level that determines where the member will be visible. There are six of them available in C#: `public`, `protected`, `internal`, `protected internal`, `private`, and `private protected`, the last of which was added in C# 7.2. The default access level for members of a class is `private`.

Private Access

All members regardless of access level are accessible in the class in which they are declared, the defining class. This is the only place where a private member can be accessed.

```
public class MyBase
{
  // Unrestricted access
  public int myPublic;

  // Defining assembly or derived class
  protected internal int myProtInt;

  // Derived class within defining assembly
  private protected int myPrivProt;
```


M. Olsson, *C# 8 Quick Syntax Reference*, https://doi.org/10.1007/978-1-4842-5577-3_13

```
  // Defining assembly
  internal int myInternal;

  // Derived class
  protected int myProtected;

  // Defining class only
  private int myPrivate;

  void Test()
  {
    myPublic    = 0; // allowed
    myProtInt   = 0; // allowed
    myPrivProt  = 0; // allowed
    myInternal  = 0; // allowed
    myProtected = 0; // allowed
    myPrivate   = 0; // allowed
  }
}
```

Protected Access

A protected member can be accessed from within a derived class, but it is inaccessible from any other classes.

```
class Derived : MyBase
{
  void Test()
  {
    myPublic    = 0; // allowed
    myProtInt   = 0; // allowed
    myPrivProt  = 0; // allowed
```

```
    myInternal  = 0; // allowed
    myProtected = 0; // allowed
    myPrivate   = 0; // inaccessible
  }
}
```

Internal Access

An internal member can be accessed anywhere within the local assembly, but not from another assembly. An assembly is the compilation unit of a .NET project, either an executable program (`.exe`) or a library (`.dll`).

```
// Defining assembly
class AnyClass
{
  void Test(MyBase m)
  {
    m.myPublic    = 0; // allowed
    m.myProtInt   = 0; // allowed
    m.myPrivProt  = 0; // inaccessible
    m.myInternal  = 0; // allowed
    m.myProtected = 0; // inaccessible
    m.myPrivate   = 0; // inaccessible
  }
}
```

In Visual Studio, a project (assembly) is contained within a solution. You can add a second project to your solution by right-clicking the Solution node in the Solution Explorer window and selecting Add ➤ New Project.

For the second project to be able to reference accessible types from the first project, you need to add a reference. To do so, right-click the References node of the second project and click Add Reference. Under Projects, select the name of the first project and click OK to add the reference.

Protected Internal Access

Protected internal access means either protected or internal. A protected internal member can therefore be accessed anywhere within the current assembly or in classes outside the assembly that are derived from the enclosing class.

```
// Other assembly
class Derived : MyBase
{
  void Test()
  {
    myPublic    = 0; // allowed
    myProtInt   = 0; // allowed
    myPrivProt  = 0; // inaccessible
    myInternal  = 0; // inaccessible
    myProtected = 0; // allowed
    myPrivate   = 0; // inaccessible
  }
}
```

Private Protected Access

A private protected member is accessible only within the defining assembly in types that derive from the defining type. Put another way, this access level restricts the member's visibility to being both protected and internal.

```
// Defining assembly
class Derived : MyBase
{
  void Test()
  {
    myPublic    = 0; // allowed
    myProtInt   = 0; // allowed
    myPrivProt  = 0; // allowed
    myInternal  = 0; // allowed
    myProtected = 0; // allowed
    myPrivate   = 0; // inaccessible
  }
}
```

Public Access

The `public` modifier gives unrestricted access from anywhere that a member can be referenced.

```
// Other assembly
class AnyClass
{
  void Test(MyBase m)
  {
    m.myPublic    = 0; // allowed
    m.myProtInt   = 0; // inaccessible
    m.myPrivProt  = 0; // inaccessible
    m.myInternal  = 0; // inaccessible
    m.myProtected = 0; // inaccessible
    m.myPrivate   = 0; // inaccessible
  }
}
```

Top-Level Access Levels

A top-level member is a type that is declared outside of any other types. In C#, the following types can be declared on the top level: `class`, `interface`, `struct`, `enum`, and `delegate`. By default, these uncontained members are given internal access. To be able to use a top-level member from another assembly, that member has to be marked as `public`. This is the only other access level allowed for top-level members.

```
internal class MyInternalClass {}
public class MyPublicClass {}
```

Inner Classes

Classes may contain inner classes, which can be set to any one of the six access levels. The access levels have the same effect on inner classes as they do on other members. If the class is inaccessible, it cannot be instantiated or inherited. By default, inner classes are private, which means that they can only be used within the class where they are defined.

```
class MyBase
{
  // Inner classes (nested classes)
  public class MyPublic {}
  protected internal class MyProtInt {}
  private protected class MyPrivProt {}
  internal class MyInternal {}
  protected class MyProtected {}
  private class MyPrivate {}
}
```

Access Level Guideline

As a guideline, when choosing an access level, it is generally best to use the most restrictive level possible. This is because the more places a member can be accessed, the more places it can be accessed incorrectly, which makes the code harder to debug. Using restrictive access levels will also make it easier to modify a class without breaking the code for any other programmers using that class.